NANO SYSTEMS CORP.®

REFERENCE CARD FOR THE

VIC-20®

MICROCOMPUTER

© 1983, SHIRLEY A. & PAUL P., and, DANIEL P. NANOS

VIC-20 is a Reg. TM of Commodore Business Machines, Inc.

KEY TO GRAPHICS NOTATION	
KEYS: ¢ Commodore, ◉ Spacebar, ⊗ Shift, ©CTRL, ● RVS ON (CTRL-9)	
UPPER CASE	KEYBOARD ENTRY
UPPER CASE	**PRINT CHR$**(nn) CODE (KEY VALUE)
UPPER CASE	OTHER **PRINT CHR$**(nn) CODE
UPPER CASE	**POKE/PEEK** CODE
	CHARACTER OR GRAPHIC
lower case	keyboard entry
lower case	**print chr$**(nn) code (key value)
lower case	other **print chr$**(nn) code
lower case	**poke/peek** code

KEYS					¢+	●¢+	¢−	●¢−	¢£	●¢£	
CHR$					166	166	220	220	168	168	
CHR$					230	230	124	124	232	232	
POKE					102	230	92	220	104	232	
KEYS	⊗£	●⊗£	¢ *	●¢*	¢+	●¢+	¢−	●¢−	¢£	●¢£	
CHR$	169	169	223	223	166	166	220	220	168	168	
CHR$	233	233	127	127	230	230	124	124	232	232	
POKE	105	233	95	223	102	230	92	220	104	232	

KEYS	¢Q	¢W	¢E	¢R	¢D	¢F	¢C	¢V	⊗£	¢ *	¢B		
CHR$	171	179	177	178	172	187	188	190	169	223	191		
CHR$	235	243	241	242	236	251	252	254	233	127			
POKE	107	115	113	114	108	123	124	126	105	95	127		
KEYS	¢Q	¢W	¢E	¢R	¢D	¢F	¢C	¢V			¢B	⊗↑	¢↑
CHR$	171	179	177	178	172	187	188	190			191	255	255
CHR$	235	243	241	242	236	251	252	254				126	222
POKE	107	115	113	114	108	123	124	126			127	94	94

KEYS	●¢Q	●¢W	●¢E	●¢R	●¢D	●¢F	●¢C	●¢V	●⊗£	●¢*	●¢B		
CHR$	171	179	177	178	172	187	188	190	169	223	191		
CHR$	235	243	241	242	236	251	252	254	233	127			
POKE	235	243	241	242	236	251	252	254	233	223	255		
KEYS	●¢Q	●¢W	●¢E	●¢R	●¢D	●¢F	●¢C	●¢V			●¢B	●⊗↑	●¢↑
CHR$	171	179	177	178	172	187	188	190			191	255	255
CHR$	235	243	241	242	236	251	252	254				126	222
POKE	235	243	241	242	236	251	252	254			255	222	222

KEYS	⊖U	⊖Y	⊖T	⊗E	⊗D	⊗C	⊗*	⊗F	⊗R	⊖@	⊖P	⊖O	⊖I
CHR$	184	183	163	197	196	195	192	198	210	164	175	185	162
CHR$	248	247	227	101	100	99	96	102	114	228	239	249	226
POKE	120	119	99	69	68	67	64	70	82	100	111	121	98
KEYS	⊖U	⊖Y	⊖T				⊗*			⊖@	⊖P	⊖O	⊖I
CHR$	184	183	163				192			164	175	185	162
CHR$	248	247	227				96			228	239	249	226
POKE	120	119	99				64			100	111	121	98

KEYS	●⊖K	⊖L	⊖N	⊖M	⊗Y	⊗H	⊗−	⊗B	⊗G	⊗T	⊖G	⊖H	⊖J
CHR$	161	182	170	167	217	200	221	194	199	212	165	180	181
CHR$	225	246	234	231	121	104	125	98	103	116	229	244	245
POKE	225	118	106	103	89	72	93	66	71	84	101	116	117
KEYS	●⊖K	⊖L	⊖N	⊖M			⊗−				⊖G	⊖H	⊖J
CHR$	161	182	170	167			221				165	180	181
CHR$	225	246	234	231			125				229	244	245
POKE	225	118	106	103			93				101	116	117

KEYS	⊗S	⊗Z	⊗A	⊗X	⊗Q	⊗W	⊗M	⊗N	⊗V	⊗+	£	↑	←
CHR$	211	218	193	216	209	215	205	206	214	219	92	94	95
CHR$	115	122	97	120	113	119	109	110	118	123			
POKE	83	90	65	88	81	87	77	78	86	91	28	30	31
KEYS										⊗+	£	↑	←
CHR$										219	92	94	95
CHR$										123			
POKE										91	28	30	31

KEYS	⊗O	⊗P	⊗L	⊗@	⊖A	⊖S	⊖Z	⊖X	⊗U	⊗I	⊗J	⊗K	
CHR$	207	208	204	186	176	174	173	189	213	201	202	203	
CHR$	111	112	108	250	240	238	237	253	117	105	106	107	
POKE	79	80	76	122	112	110	109	125	85	73	74	75	
KEYS					⊖A	⊖S	⊖Z	⊖X					
CHR$					176	174	173	189					
CHR$					240	238	237	253					
POKE					112	110	109	125					

KEYS	!	”	#	$	%	&	’	(	)	*	+	,	−
CHR$	33	34	35	36	37	38	39	40	41	42	43	44	45
CHR$													
POKE	33	34	35	36	37	38	39	40	41	42	43	44	45
KEYS	!	”	#	$	%	&	’	(	)	*	+	,	−
CHR$	33	34	35	36	37	38	39	40	41	42	43	44	45
CHR$													
POKE	33	34	35	36	37	38	39	40	41	42	43	44	45

KEYS	.	:	;	/	=	?	@	<	>	[	]	⊗↑	⊖↑
CHR$	46	58	59	47	61	63	64	60	62	91	93	255	255
CHR$												222	126
POKE	46	58	59	47	61	63	0	60	62	27	29	94	94
KEYS	.	:	;	/	=	?	@	<	>	[	]		
CHR$	46	58	59	47	61	63	64	60	62	91	93		
CHR$													
POKE	46	58	59	47	61	63	0	60	62	27	29		

KEYS	●⊖U	●⊖Y	●⊖T	●⊗E	●⊗D	●⊗C	●⊗*	●⊗F	●⊗R	●⊖@	●⊖P	●⊖O	●⊖I
CHR$	184	183	163	197	196	195	192	198	210	164	175	185	162
CHR$	248	247	227	101	100	99	96	102	114	228	239	249	226
POKE	248	247	227	197	196	195	192	198	210	228	239	249	226
KEYS	●⊖U	●⊖Y	●⊖T				⊗*			●⊖@	●⊖P	●⊖O	●⊖I
CHR$	184	183	163				192			164	175	185	162
CHR$	248	247	227				96			228	239	249	226
POKE	248	247	227				192			228	239	249	226

KEYS	⊖K	⊖L	⊖N	⊖M	⊗Y	⊗H	⊗−	⊗B	⊗G	⊗T	●⊖G	●⊖H	●⊖J
CHR$	161	182	170	167	217	200	221	194	199	212	165	180	181
CHR$	225	246	234	231	121	104	125	98	103	116	229	244	245
POKE	97	246	234	231	217	200	221	194	199	212	229	244	245
KEYS	⊖K	⊖L	⊖N	⊖M			⊗−				●⊖G	●⊖H	●⊖J
CHR$	161	182	170	167			221				165	180	181
CHR$	225	246	234	231			125				229	244	245
POKE	97	246	234	231			221				229	244	245

KEYS	●⊗S	●⊗Z	●⊗A	●⊗X	●⊗Q	●⊗W	●⊗M	●⊗N	●⊗V	●⊗+	●£	●↑	●←
CHR$	211	218	193	216	209	215	205	206	214	219	92	94	95
CHR$	115	122	97	120	113	119	109	110	118	123			
POKE	211	218	193	216	209	215	205	206	214	219	156	158	159
KEYS										●⊗+	●£	●↑	●←
CHR$										219	92	94	95
CHR$										123			
POKE										219	156	158	159

KEYS	●⊗O	●⊗P	●⊗L	●⊗@	●⊖A	●⊖S	●⊖Z	●⊖X	●⊗U	●⊗I	●⊗J	●⊗K	
CHR$	207	208	204	186	176	174	173	189	213	201	202	203	
CHR$	111	112	108	250	240	238	237	253	117	105	106	107	
POKE	207	208	204	250	240	238	237	253	213	201	202	203	
KEYS					●⊖A	●⊖S	●⊖Z	●⊖X					
CHR$					176	174	173	189					
CHR$					240	238	237	253					
POKE					240	238	237	253					

KEYS	●!	●”	●#	●$	●%	●&	●’	●(	●)	●*	●+	●,	●−
CHR$	33	34	35	36	37	38	39	40	41	42	43	44	45
CHR$													
POKE	161	162	163	164	165	166	167	168	169	170	171	172	173
KEYS	●!	●”	●#	●$	●%	●&	●’	●(	●)	●*	●+	●,	●−
CHR$	33	34	35	36	37	38	39	40	41	42	43	44	45
CHR$													
POKE	161	162	163	164	165	166	167	168	169	170	171	172	173

KEYS	●.	●:	●;	●/	●=	●?	●@	●<	●>	●[	●]	⊗↑	⊖↑
CHR$	46	58	59	47	61	63	64	60	62	91	93	255	255
CHR$												222	126
POKE	174	186	187	175	189	191	128	188	190	155	157	222	222
KEYS	●.	●:	●;	●/	●=	●?	●@	●<	●>	●[	●]		
CHR$	46	58	59	47	61	63	64	60	62	91	93		
CHR$													
POKE	174	186	187	175	189	191	128	188	190	155	157		

KEYS	0	1	2	3	4	5	6	7	8	9		◉	⊗◉
CHR$	48	49	50	51	52	53	54	55	56	57		32	160
CHR$													224
POKE	48	49	50	51	52	53	54	55	56	57		32	96
	0	1	2	3	4	5	6	7	8	9	✓		

KEYS	0	1	2	3	4	5	6	7	8	9	⊗@	◉	⊗◉
CHR$	48	49	50	51	52	53	54	55	56	57	186	32	160
CHR$											250		224
POKE	48	49	50	51	52	53	54	55	56	57	122	32	96

KEYS	A	B	C	D	E	F	G	H	I	J	K	L	M
CHR$	65	66	67	68	69	70	71	72	73	74	75	76	77
CHR$													
POKE	1	2	3	4	5	6	7	8	9	10	11	12	13
	A	B	C	D	E	F	G	H	I	J	K	L	M

KEYS	⊗A	⊗B	⊗C	⊗D	⊗E	⊗F	⊗G	⊗H	⊗I	⊗J	⊗K	⊗L	⊗M
CHR$	193	194	195	196	197	198	199	200	201	202	203	204	205
CHR$	97	98	99	100	101	102	103	104	105	106	107	108	109
POKE	65	66	67	68	69	70	71	72	73	74	75	76	77

KEYS	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
CHR$	78	79	80	81	82	83	84	85	86	87	88	89	90
CHR$													
POKE	14	15	16	17	18	19	20	21	22	23	24	25	26
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

KEYS	⊗N	⊗O	⊗P	⊗Q	⊗R	⊗S	⊗T	⊗U	⊗V	⊗W	⊗X	⊗Y	⊗Z
CHR$	206	207	208	209	210	211	212	213	214	215	216	217	218
CHR$	110	111	112	113	114	115	116	117	118	119	120	121	122
POKE	78	79	80	81	82	83	84	85	86	87	88	89	90

KEYS	NONE												
CHR$	NONE												
CHR$	NONE												
POKE	NONE												
	a	b	c	d	e	f	g	h	i	j	k	l	m
KEYS	A	B	C	D	E	F	G	H	I	J	K	L	M
CHR$	65	66	67	68	69	70	71	72	73	74	75	76	77
CHR$													
POKE	1	2	3	4	5	6	7	8	9	10	11	12	13

KEYS	NONE												
CHR$	NONE												
CHR$	NONE												
POKE	NONE												
	n	o	p	q	r	s	t	u	v	w	x	y	z
KEYS	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
CHR$	78	79	80	81	82	83	84	85	86	87	88	89	90
CHR$													
POKE	14	15	16	17	18	19	20	21	22	23	24	25	26

KEYS	●0	●1	●2	●3	●4	●5	●6	●7	●8	●9		●◉	●⊗◉
CHR$	48	49	50	51	52	53	54	55	56	57		32	160
CHR$													224
POKE	176	177	178	179	180	181	182	183	184	185		160	224
	0	1	2	3	4	5	6	7	8	9	✓		

KEYS	●0	●1	●2	●3	●4	●5	●6	●7	●8	●9	●⊗@	●◉	●⊗◉
CHR$	48	49	50	51	52	53	54	55	56	57	186	32	160
CHR$											250		224
POKE	176	177	178	179	180	181	182	183	184	185	250	160	224

KEYS	●A	●B	●C	●D	●E	●F	●G	●H	●I	●J	●K	●L	●M
CHR$	65	66	67	68	69	70	71	72	73	74	75	76	77
CHR$													
POKE	129	130	131	132	133	134	135	136	137	138	139	140	141
	A	B	C	D	E	F	G	H	I	J	K	L	M

KEYS	●⊗A	●⊗B	●⊗C	●⊗D	●⊗E	●⊗F	●⊗G	●⊗H	●⊗I	●⊗J	●⊗K	●⊗L	●⊗M
CHR$	193	194	195	196	197	198	199	200	201	202	203	204	205
CHR$	97	98	99	100	101	102	103	104	105	106	107	108	109
POKE	193	194	195	196	197	198	199	200	201	202	203	204	205

KEYS	●N	●O	●P	●Q	●R	●S	●T	●U	●V	●W	●X	●Y	●Z
CHR$	78	79	80	81	82	83	84	85	86	87	88	89	90
CHR$													
POKE	142	143	144	145	146	147	148	149	150	151	152	153	154
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

KEYS	●⊗N	●⊗O	●⊗P	●⊗Q	●⊗R	●⊗S	●⊗T	●⊗U	●⊗V	●⊗W	●⊗X	●⊗Y	●⊗Z
CHR$	206	207	208	209	210	211	212	213	214	215	216	217	218
CHR$	110	111	112	113	114	115	116	117	118	119	120	121	122
POKE	206	207	208	209	210	211	212	213	214	215	216	217	218

KEYS	NONE												
CHR$	NONE												
CHR$	NONE												
POKE	NONE												
	a	b	c	d	e	f	g	h	i	j	k	l	m
KEYS	●A	●B	●C	●D	●E	●F	●G	●H	●I	●J	●K	●L	●M
CHR$	65	66	67	68	69	70	71	72	73	74	75	76	77
CHR$													
POKE	129	130	131	132	133	134	135	136	137	138	139	140	141

KEYS	NONE												
CHR$	NONE												
CHR$	NONE												
POKE	NONE												
	n	o	p	q	r	s	t	u	v	w	x	y	z
KEYS	●N	●O	●P	●Q	●R	●S	●T	●U	●V	●W	●X	●Y	●Z
CHR$	78	79	80	81	82	83	84	85	86	87	88	89	90
CHR$													
POKE	142	143	144	145	146	147	148	149	150	151	152	153	154

MUSICAL SCALE

NOTE	POKE VALUE							
C	133	194	225	240				
C#	139	198	226	241	248			
D	146	201	228	242				
D#	151	203	230					
E	157	206	231	243	249	252		
F	163	209	232	244				
F#	168	212	233					
G	174	215	235	245	250			
G#	178	217	236					
A	183	219	237	246				
A#	187	221	238					
B	128	190	223	239	247	251	253	255

SPEAKER 1 = 36874, Lowest tones. Example: POKE 36874,163
SPEAKER 2 = 36875, Middle tones. Example: POKE 36875,250
SPEAKER 3 = 36876, Higher tones. Example: POKE 36876,183
SPEAKER 4 = 36877, White noises. Example: POKE 36877,135
VOLUME CONTROL = 36878, (0-15). Example: POKE 36878,15

GRAPHIC PRINTER CONTROL CODES

DEC	HEX	FUNCTION
8	08	Enter GRAPHICS mode.
10	0A	Feed paper up 1 line. (LF, linefeed).
13	0D	Return carriage to start of line. (CR, carriage return)
14	0E	Turn on elongated character mode.
15	0F	Exit GRAPHICS mode. Turn off elongated char mode.
16	10	Prepare for print position addressing.
17	11	Select lower case character set.
18	12	Turn on reverse character mode.
26	1A	Prepare for repeat graphics.
27,16	1B,10	Prepare for print dot position addressing.
145	91	Select upper case character set.
146	92	Turn off reverse character mode.

To print your own 7 × 7 graphic, open your printer, enter GRAPHICS mode, then print your graphic. Below is an example of how the codes for a new graphic are made. Add up the values in each column of the boxes with your design in them, then add 128 to each column. Then assign a variable to the string of chr$ resulting, and, the variable contains the data for your new GRAPHIC character which you can print.

BITS							VALUE
1	1	1	1	1	0	0	1
1	1	1	0	1	0	0	2
0	0	1	1	1	1	1	4
0	0	1	1	1	1	1	8
1	1	1	0	0	0	0	16
1	1	1	1	1	0	0	32
0	0	1	1	0	0	0	64
COLUMN TOTAL	51	51	127	109	111	12	12
PLUS 128 =	179	179	255	237	239	140	140

```
A$=chr$(179)+chr$(179)+chr$(255)+chr$(237)+chr$(239)+chr$(140)+chr$(140)
PRINT#dev,A$
```


OPEN STATEMENT COMMAND DATA CODES (OPTIONS)

CASSETTE TAPE

0 Open for read only.
1 Open for write.
2 Open for write. Write END-OF-TAPE (EOT) at end.

DISK DRIVE

1—14 Open data channel.
15 Open command channel.

HARD COPY PRINTER

0 Upper—case & graphics set.
7 Upper & lower case set.

RS—232 INTERFACE CODES are varied depending upon your usage. They set STOP BITS, WORD LENGTH, BAUD RATE, PARITY, DUPLEX, HANDSHAKE.

HEX/DEC CONVERSION CHART

8 4 2 1		8 4 2 1		8 4 2 1		8 4 2 1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15
65535		4095		255		15	

BASIC SPECIAL CHARACTERS AND OPERATORS

?	Use instead of "PRINT". (?A$)
blank	End of BASIC statement. Tab to next line after PRINT.
:	End of BASIC statement. Tab to next line after PRINT.
,	Tab to column 1 or 12 after PRINT.
;	Suppress Tab after PRINT.
" "	Delimits a character string. ("SAM")
$	Identifies a string variable (text). (A$="SAM")
%	Identifies an integer variable (numeric). (A%=123)
blank	Identifies a floating point variable (numeric). (A=1.23)
E	Floating point notation. 1E9=One and 9 zeros. 1E−9=Decimal, 8 zeros, and one.
( )	Top priority in order of operations. ((3*4)/ 2=6)
↑	Exponentiation (power). (2↑3=8)
*****	Multiplication. (4*5=20)
/	Division. (6/ 2=3)
+	Addition, or concatentation of strings (head-to-toe). (4+3=7. A$+B$=C$)
−	Negative or subtraction. (−3−2=−5)
=	Equals, or assignment. (A=3. (If A=3)
>	Greater than. (If A>B)
<	Less than. (If B<A)
>= or =>	Greater than or equal to. (If A>=B)
<= or =<	Less than or equal to. (If B<=A)
<> or ><	Not equal to. (If A<>B)
NOT	Reverses True/ False result. (If A NOT >B)
AND	If both expressions are true, result is true. (If exp AND exp)
OR	If either or both expressions are true, result is true. (if exp OR exp)
Non-zero	True result.
Zero	False result.

Order of operations:
( ), {NOT, −(neg), +(pos)}, ↑, {*, /}, {+, −}, {>, <, =, >=, <=, <>}, AND, OR

SCREEN LINE LAYOUT

( ) = 16K, 24K, 32K Systems Only

LINE NO.	DISPLACE DEC	DISPLACE HEX	GRAPHIC LOCATION DEC	GRAPHIC LOCATION HEX	COLOR LOCATION DEC	COLOR LOCATION HEX
0	0	0000	7680 (4096)	1E00 (1000)	38400 (37888)	9600 (9400)
1	22	0016	7702 (4118)	1E16 (1016)	38422 (37910)	9616 (9416)
2	44	002C	7724 (4140)	1E2C (102C)	38444 (37932)	962C (942C)
3	66	0042	7746 (4162)	1E42 (1042)	38466 (37954)	9642 (9442)
4	88	0058	7768 (4184)	1E58 (1058)	38488 (37976)	9658 (9458)
5	110	006E	7790 (4206)	1E6E (106E)	38510 (37998)	966E (946E)
6	132	0084	7812 (4228)	1E84 (1084)	38532 (38020)	9684 (9484)
7	154	009A	7834 (4250)	1E9A (109A)	38554 (38042)	969A (949A)
8	176	00B0	7856 (4272)	1EB0 (10B0)	38576 (38064)	96B0 (94B0)
9	198	00C6	7878 (4294)	1EC6 (10C6)	38598 (38086)	96C6 (94C6)
10	220	00DC	7900 (4316)	1EDC (10DC)	38620 (38108)	96DC (94DC)
11	242	00F2	7922 (4338)	1EF2 (10F2)	38642 (38130)	96F2 (94F2)
12	264	0108	7944 (4360)	1F08 (1108)	38664 (38152)	9708 (9508)
13	286	011E	7966 (4382)	1F1E (111E)	38686 (38174)	971E (951E)
14	308	0134	7988 (4404)	1F34 (1134)	38708 (38196)	9734 (9534)
15	330	014A	8010 (4426)	1F4A (114A)	38730 (38218)	974A (954A)
16	352	0160	8032 (4448)	1F60 (1160)	38752 (38240)	9760 (9560)
17	374	0176	8054 (4470)	1F76 (1176)	38774 (38262)	9776 (9576)
18	396	018C	8076 (4492)	1F8C (118C)	38796 (38284)	978C (958C)
19	418	01A2	8098 (4514)	1FA2 (11A2)	38818 (38306)	97A2 (95A2)
20	440	01B8	8120 (4536)	1FB8 (11B8)	38840 (38328)	97B8 (95B8)
21	462	01CE	8142 (4558)	1FCE (11CE)	38862 (38350)	97CE (95CE)
22	484	01E4	8164 (4580)	1FE4 (11E4)	38884 (38372)	97E4 (95E4)
END	505	01F9	8185 (4601)	1FF9 (11F9)	38905 (38393)	97F9 (95F9)

Poke 7680-8185 (4096-4601) with number of graphic desired at that location.
Peek 7680-8185 (4096-4601) to see number of graphic at that location.
(Graphic codes are from 0-255)
Poke 38400-38905 (37888-38393) with number of color desired for graphic at that location.
Peek 38400-38905 (37888-38393) to see number of graphic color at that location.
(Color codes are from 0-7, same as keyboard colors 1-8)
0-Black, 1-White, 2-Red, 3-Cyan, 4-Purple, 5-Green, 6-Blue, 7-Yellow

SETTING SCREEN & BORDER COLORS

Assign your codes as follows: (Default = 27)

	BORDER COLOR	BORDER CODE		SCREEN COLOR	SCREEN CODE
	Black	8		Black	0
	White	9		White	16
	Red	10		Red	32
	Cyan	11		Cyan	48
	Purple	12		Purple	64
X=	Green	13	+	Green	80
	Blue	14		Blue	96
	Yellow	15		Yellow	112
				Orange	128
				Lt Orange	144
				Pink	180
				Lt Cyan	176
				Lt Purple	192
				Lt Green	208
				Lt Blue	224
				Lt Yellow	240

X=Border code plus Screen code.

Poke 36879,X ..... Sets display to these colors.
Poke 36879,X−8 ... Sets border to color selected, forces screen color to white, reverses characters on the display, and turns character colors to same color as screen code used.

SPECIAL KEYBOARD KEYS

KEYS: ¢ Commodore, ◉ Spacebar, ⊗ Shift, ©CTRL

UP	LOW	DEC	HEX	KEY	RESULT
				©	*Slow down screen display.*
				⊗	*Use for upper-case or right-side graphics.*
				¢	*Use for left-side graphics.*
				¢⊗	*Toggle upper/ lower case char. sets.*
		144	90	©1	*Set char. color to black.*
		5	5	©2	*Set char. color to white.*
		28	1C	©3	*Set char. color to red.*
		159	9F	©4	*Set char. color to cyan.*
		156	9C	©5	*Set char. color to purple.*
		30	1E	©6	*Set char. color to green.*
		31	1F	©7	*Set char. color to blue.*
		158	9E	©8	*Set char. color to yellow.*
		6	6	©←	
		18	12	©9	*Set reverse on.*
		18	12	©R	*Set reverse on.*
		146	92	©0	*Set reverse off.*
		145	91	CRSR ↑	*Move cursor up.*
		29	1D	CRSR →	*Move cursor right.*
		157	9D	CRSR ←	*Move cursor left.*
		17	11	CRSR ↓	*Move cursor down.*
		147	93	CLR	*Clear screen, move cursor to upper left.*
		20	14	DEL	*Delete character.*
		133	85	F1	*User function key 1.*
		137	89	F2	*User function key 2.*
		134	86	F3	*User function key 3.*
		138	8A	F4	*User function key 4.*
		135	87	F5	*User function key 5.*
		139	8B	F6	*User function key 6.*
		136	88	F7	*User function key 7.*
		140	8C	F8	*User function key 8.*
		197	C5	HOME	*Move cursor to upper left.*
		148	94	INST	*Insert character.*
				RESTORE	*When pressed simultaneously with RUN/ STOP, resets computer, keeps program in memory.*
		13	0D	RETURN	*Enter data.*
		141	8D	⊗RETURN	*Move cursor to next line.*
LOAD		131	83	RUN	*Load program from cassette.*
				STOP	*Break, halt execution.*

INPUT/OUTPUT DEVICE (dev) IDENTIFICATION CODES

0	Computer Keyboard
1	Cassette Tape
2	RS232 Interface Device
3	Screen Display
4	Hard-Copy Printer
5	Hard-Copy Printer 2
8	Disk Unit
4−127	Serial-Bus Device (Carriage Return Only)
128−255	Serial-Bus Device (Carriage Return/Line-Feed)

INPUT/OUTPUT STATUS CHART

STATUS CODE	REASON
1	Serial bus time-out, write.
2	Serial bus time-out, read.
4	Tape−short block.
8	Tape−long block.
16	Tape−read error or bad compare.
32	Tape−checksum error.
64	Tape−end-of-file. Serial bus−EOI.
−128	Tape−end-of-tape. Serial bus−device not present.

BASIC COMMANDS

EXAMPLE	DESCRIPTION
CONT	CONTinue execution after break, **STOP**, or **END**.
LIST	Display all **BASIC** program lines.
LIST 10	Display **BASIC** program line 10.
LIST 10—100	Display **BASIC** lines 10 thru 100.
LIST 200—	Display **BASIC** lines 200 thru end of program.
LIST —150	Display **BASIC** lines 0 thru 150.
LOAD	Load first program on tape into memory. (When used inside **BASIC** program, executes at once.
LOAD "TEST"	Load program named **TEST** into memory from tape.
LOAD "TEST",dev	Load program named **TEST** into memory from device. (dev = 1 for tape, 8 for disk).
LOAD "TEST",dev,1	Load program into same location saved from.
LOAD " * ",8	Load first program on disk into memory.
LOAD "TE * ",8	Load first program, prefix TE to memory from disk.
LOAD A$	Load program named in variable A$ from tape.
NEW	Clear current program out of memory.
RUN	Execute **BASIC** program.
RUN 1000	Execute **BASIC** program starting at line 1000.
SAVE	Save program in memory onto tape without name.
SAVE "TEST"	Save program onto tape, and name it **TEST**.
SAVE A$	Save program onto tape, using name in variable A$.
SAVE A$,dev	Save program onto device (1-tape, 8-disk).
SAVE "TEST",dev,opt	Save program onto device with option: 1—Record program's address in memory for loading. 2—Place an end-of-tape record after the program. 3—Do both options 1 and 2 above.
VERIFY	Compare first program on tape against memory program
VERIFY "TEST"	Compare tape program **TEST** against one in memory.
VERIFY "TEST",8	Compare disk program **TEST** against one in memory.

BASIC STATEMENTS [ ] = Optional

STATEMENT • DESCRIPTION

CLOSE fileno • Close open file **fileno.**
CLR • Clear vars & arrays, and **RESTORE** pointer to **DATA** table.
CMD fileno • Route screen output to **fileno** until **PRINT#** and **CLOSE.**
DATA [item [,item . . .] • Store **item**s into data memory table.
DEF FNx(arg)=expression • Define user function named **FNx.**
DIM var (n1 [,n2, . . .]) [,var1(n3,n4, . . .)] • Define array storage.
END • Halt program execution until **CONT.**
FOR var=num1 TO num2 [STEP incr] • Loop thru **NEXT** until **var** greater **num2.**
GET var$** • Scan keyboard and place key pressed into **var$.
GET# fileno, var • Read byte from **fileno** into **var.**
GOSUB lineno • Perform routine at **lineno** until **RETURN.**
GOTO lineno • Branch or transfer control to **lineno.**
IF cond THEN lineno • If **cond**ition met then branch to **lineno.**
IF cond THEN action [:action . . .] • If **cond**ition met then do **action**(s).
INPUT [msg;] var [,var . . .] • Prompt for and read keyed response.
INPUT# fileno, var [,var . . .] • Read data from **fileno** into **var**(s).
[LET] var=value • Assign a **value** to a **var**iable.
NEXT [var [,var . . .] • Identify end of **FOR** loop(s).
ON num GOSUB line [,line . . .] • Perform routine **num** of **line** list.
ON num GOTO line [,line . . .] • Branch to lineno **num** of **line** list.
OPEN fileno [,**dev**] [,**option**] [,**data**] • **Open fileno on device dev using option and filename/param data.**
POKE addr,value • Move **value** (0—255) to **addr** in memory.
PRINT item [; [, [item . . .] • Display **item**(s) on screen. (TAB & SPC useable)
PRINT# fileno,item [; [, [item . . .] • Write **item**(s) onto **fileno.**
READ var [,var . . .] • Read **DATA** from memory table.
REM • **REM**arks follow this statement.
RESTORE • Reset pointer to start of **DATA** memory table.
RETURN • Return from **GOSUB.**
STOP • Halt program execution until **CONT.**
SYS addr • Branch to machine language routine at **addr.**
TI$=num • Set clock timer (HHMMSS). Hours, minutes, seconds.
WAIT addr, and1 [,xor2] • Pause execution until data at **addr** matches arg.

Published 1983 by Nanos Systems Corp., P.O. Box 24344, Speedway, IN 46224
(317) 244 - 4078

Printed in U.S.A.

BASIC FUNCTIONS

FUNCTION	RETURNS
ABS(num)	Absolute value of **num**. (—3.25=3.25)
ASC(str)	Decimal value of char. ("A"=65)
ATN(num)	Angle (in radians) whose tangent is **num.**
CHR$(num)	Character value of **num**. (65="A")
COS(angle-rads)	Cosine (in radians) of angle radians.
EXP(num)	Inverse LOG of **num** (2.71827183 to power of **num**).
FNx(arg)	User function result. (See DEF FN)
FRE(any-num)	Number of available bytes left in memory.
INT(num)	Largest whole number not exceeding **num.** (2.5=2) (—2.5=—3)
INT(RND(1)*num)+x	Random whole number between **x**—1 and **x**+**num.**
LEFT$(str,num)	Substring of **num** chars from left of **str**ing.
LEN(str)	Length of **str**ing.
LOG(num)	Natural logaritm of **num.**
MID$(str,pos[,len])	Substring of **str**, position **pos**, length **len.**
PEEK(addr)	Decimal value of byte at location **addr.**
POS(any-num)	Cursor position on screen. Column (0—21).
RIGHT$(str,num)	Substring of **num** chars from right of **str**ing.
RND(any-num)	Random number between 0 and 1.
RND(neg-num)	Re-seed random generator. (Same seed if same num)
RND(0)	Last random number generated.
SGN(num)	0(zero), 1(positive), —1(negative).
SIN(angle-rads)	Sine (in radians) of angle radians.
SPC(num)	In a PRINT statement, **num** spaces.
SQR(positive num)	Square root of a **positive num**ber.
ST	Status code from last input/output operation. See INPUT/OUTPUT STATUS CHART
STR$(num)	Character string of **num**, including its sign.
TAB(col)	In a PRINT statement, tab to column **col.**
TAN(angle-rads)	Tangent (in radians) of angle radians.
TI	Time, in 1/10th seconds.
TI$	Time, hhmmss (hours, minutes, seconds).
USR(parm)	Branch to machine language routine whose starting point is in locations 1 and 2, passing parameter **parm** to it.
VAL(str)	Numeric value of number inside of **str**ing.
π	Pi. (3.14159265).

DERIVED FUNCTIONS (X is in radians.)

FUNCTION	EXPRESSION
SECANT	1/COS(X)
COSECANT	1/SIN(X)
COTANGENT	1/TAN(X)
INVERSE	
SINE	ATN(X/SQR(—X*X + 1))
COSINE	—ATN(X/SQR(—X*X + 1)) + 1.570796
SECANT	ATN(SQR(X*X — 1)) + (SGN(X) —1) *1.570796
COSECANT	ATN(1/SQR(X*X —1)) + (SGN(X) —1)*1.570796
COTANGENT	—ATN(X) + 1.570796
HYPERBOLIC	
SINE	(EXP(X) — EXP(—X))/2
COSINE	(EXP(X) + EXP(—X))/2
TANGENT	—EXP(—X)/(EXP(X) + EXP(—X))*2 + 1
SECANT	2/(EXP(X) + EXP(—X))
COSECANT	2/(EXP(X) — EXP(—X))
COTANGENT	EXP(—X)/(EXP(X) — EXP(—X))*2 + 1
INVERSE HYPERBOLIC	
SINE	LOG(X + SQR(X*X + 1))
COSINE	LOG(X + SQR(X*X — 1))
TANGENT	LOG((1 +X)/(1 —X))/2
SECANT	LOG((SQR(—X*X + 1) + 1)/X)
COSECANT	LOG((SGN(X)*SQR(X*X + 1) + 1)/X)
COTANGENT	LOG((X + 1)/(X — 1))/2

DEC	HEX	GET KEY	UPPER CASE PRINT CHR$	UPPER CASE POKE	LOWER CASE PRINT CHR$	LOWER CASE POKE	TOKEN OR ASCII
0	00			@		@	NUL
1	01			A		a	SOH
2	02			B		b	STX
3	03			C		c	ETX
4	04			D		d	EOT
5	05	©2	white	E	white	e	ENQ
6	06	©←		F		f	ACK
7	07			G		g	BEL
8	08		toggle off	H	toggle off	h	BS
9	09		toggle on	I	toggle on	i	HT
10	0A			J		j	LF
11	0B			K		k	VT
12	0C			L		l	FF
13	0D	RETURN	curs next lin	M	curs next lin	m	CR
14	0E		set low case	N	set low case	n	SO
15	0F			O		o	SI
16	10			P		p	DLE
17	11	CUR↓	cursor down	Q	cursor down	q	DC1
18	12	©9 ©R	reverse on	R	reverse on	r	DC2
19	13	HOME	home cursor	S	home cursor	s	DC3
20	14	DEL	delete char	T	delete char	t	DC4
21	15			U		u	NAK
22	16			V		v	SYN
23	17			W		w	ETB
24	18			X		x	CAN
25	19			Y		y	EM
26	1A			Z		z	SUB
27	1B			[		{	ESC
28	1C	©3	red	£	red	£	FS
29	1D	CUR→	cursor right	]	cursor right	}	GS
30	1E	©6	green	↑	green	~	RS
31	1F	©7	blue	←	blue	←	US
32	20	space					space
33	21	!	!	!	!	!	!
34	22	”	"	"	"	"	”
35	23	#	#	#	#	#	#
36	24	$	$	$	$	$	$
37	25	%	%	%	%	%	%
38	26	&	&	&	&	&	&
39	27	’	’	’	’	’	’
40	28	(	(	(	(	(	(
41	29	)	)	)	)	)	)
42	2A	*	*	*	*	*	*
43	2B	+	+	+	+	+	+
44	2C	,	,	,	,	,	,
45	2D	—	—	—	—	—	—
46	2E	.	.	.	.	.	.
47	2F	/	/	/	/	/	/
48	30	0 ⊗0	0	0	0	0	0
49	31	1	1	1	1	1	1
50	32	2	2	2	2	2	2
51	33	3	3	3	3	3	3
52	34	4	4	4	4	4	4
53	35	5	5	5	5	5	5
54	36	6	6	6	6	6	6
55	37	7	7	7	7	7	7
56	38	8	8	8	8	8	8
57	39	9	9	9	9	9	9
58	3A	:	:	:	:	:	:
59	3B	;	;	;	;	;	;
60	3C	<	<	<	<	<	<
61	3D	=	=	=	=	=	=
62	3E	>	>	>	>	>	>
63	3F	?	?	?	?	?	?

DEC	HEX	GET KEY	UPPER CASE PRINT CHR$	UPPER CASE POKE	LOWER CASE PRINT CHR$	LOWER CASE POKE	TOKEN OR ASCII
64	40	@					@
65	41	A					A
66	42	B					B
67	43	C					C
68	44	D					D
69	45	E					E
70	46	F					F
71	47	G					G
72	48	H					H
73	49	I					I
74	4A	J					J
75	4B	K					K
76	4C	L					L
77	4D	M					M
78	4E	N					N
79	4F	O					O
80	50	P					P
81	51	Q					Q
82	52	R					R
83	53	S					S
84	54	T					T
85	55	U					U
86	56	V					V
87	57	W					W
88	58	X					X
89	59	Y					Y
90	5A	Z					Z
91	5B	[					[
92	5C	£					\
93	5D	]					[
94	5E	↑					^
95	5F	← ⊗←					—
96	60						‘
97	61						a
98	62						b
99	63						c
100	64						d
101	65						e
102	66						f
103	67						g
104	68						h
105	69						i
106	6A						j
107	6B						k
108	6C						l
109	6D						m
110	6E						n
111	6F						o
112	70						p
113	71						q
114	72						r
115	73						s
116	74						t
117	75						u
118	76						v
119	77						w
120	78						x
121	79						y
122	7A						z
123	7B						{
124	7C						|
125	7D						}
126	7E						~
127	7F						DEL

DEC	HEX	GET KEY	UPPER CASE PRINT CHR$	UPPER CASE POKE	LOWER CASE PRINT CHR$	LOWER CASE POKE	TOKEN OR ASCII
128	80						END
129	81						FOR
130	82						NEXT
131	83	RUN					DATA
132	84						INPUT#
133	85	F1					INPUT
134	86	F3					DIM
135	87	F5					READ
136	88	F7					LET
137	89	F2					GOTO
138	A8	F4					RUN
139	8B	F6					IF
140	8C	F8					RESTORE
141	8D	⊗RETURN	curs next lin		curs next lin		GOSUB
142	8E		set upp case		set upp case		RETURN
143	8F						REM
144	90	©1	black		black		STOP
145	91	CUR↑	cursor up		cursor up		ON
146	92	©0	reverse off		reverse off		WAIT
147	93	CLR	clear screen		clear screen		LOAD
148	94	INST	insert space		insert space		SAVE
149	95						VERIFY
150	96						DEF
151	97						POKE
152	98						PRINT#
153	99						PRINT
154	9A						CONT
155	9B						LIST
156	9C	©5	purple		purple		CLR
157	9D	CUR←	cursor left		cursor left		CMD
158	9E	©8	yellow		yellow		SYS
159	9F	©4	cyan		cyan		OPEN
160	A0	⊗space					CLOSE
161	A1	€K					GET
162	A2	€I					NEW
163	A3	€T					TAB(
164	A4	€@					TO
165	A5	€G					FN
166	A6	€+					SPC(
167	A7	€M					THEN
168	A8	€£					NOT
169	A9	⊗£					STEP
170	AA	€N					+
171	AB	€Q					−
172	AC	€D					.
173	AD	€Z					/
174	AE	€S					|
175	AF	€P					AND
176	B0	€A					OR
177	B1	€E					>
178	B2	€R					=
179	B3	€W					<
180	B4	€H					SGN
181	B5	€J					INT
182	B6	€L					ABS
183	B7	€Y					USR
184	B8	€U					FRE
185	B9	€O					POS
186	BA	⊗@					SQR
187	BB	€F					RND
188	BC	€C					LOG
189	BD	€X					EXP
190	BE	€V					COS
191	BF	€B					SIN

FUNCTION	BASIC FUNCTIONS RETURNS
ABS(num)	Absolute value of **num**. (−3.25=3.25)
ASC(str)	Decimal value of char. ("A"=65)
ATN(num)	Angle (in radians) whose tangent is **num.**
CHR$(num)	Character value of **num**. (65="A")
COS(angle-rads)	Cosine (in radians) of angle radians.
EXP(num)	Inverse LOG of **num** (2.71827183 to power of **num**).
FNx(arg)	User function result. (See DEF FN)
FRE(any-num)	Number of available bytes left in memory.
INT(num)	Largest whole number not exceeding **num.** (2.5=2) (−2.5=−3)
INT(RND(1)*num)+x	Random whole number between **x**−1 and **x**+**num.**
LEFT$(str,num)	Substring of **num** chars from left of **str**ing.
LEN(str)	Length of **str**ing.
LOG(num)	Natural logaritm of **num.**
MID$(str,pos[,len])	Substring of **str**, position **pos**, length **len.**
PEEK(addr)	Decimal value of byte at location **addr.**
POS(any-num)	Cursor position on screen. Column (0−21).
RIGHT$(str,num)	Substring of **num** chars from right of **str**ing.
RND(any-num)	Random number between 0 and 1.
RND(neg-num)	Re-seed random generator. (Same seed if same num)
RND(0)	Last random number generated.
SGN(num)	0(zero), 1(positive), −1(negative).
SIN(angle-rads)	Sine (in radians) of angle radians.
SPC(num)	In a PRINT statement, **num** spaces.
SQR(positive num)	Square root of a **positive num**ber.
ST	Status code from last input/output operation. See INPUT/OUTPUT STATUS CHART
STR$(num)	Character string of **num**, including its sign.
TAB(col)	In a PRINT statement, tab to column **col.**
TAN(angle-rads)	Tangent (in radians) of angle radians.
TI	Time, in 1/10th seconds.
TI$	Time, hhmmss (hours, minutes, seconds).
USR(parm)	Branch to machine language routine whose starting point is in locations 1 and 2, passing parameter **parm** to it.
VAL(str)	Numeric value of number inside of **str**ing.
π	Pi. (3.14159265).

DERIVED FUNCTIONS (X is in radians.)	
FUNCTION	**EXPRESSION**
SECANT	1/COS(X)
COSECANT	1/SIN(X)
COTANGENT	1/TAN(X)
INVERSE	
SINE	ATN(X/SQR(−X*X + 1))
COSINE	−ATN(X/SQR(−X*X + 1)) + 1.570796
SECANT	ATN(SQR(X*X − 1)) + (SGN(X) −1) *1.570796
COSECANT	ATN(1/SQR(X*X −1)) + (SGN(X) −1)*1.570796
COTANGENT	−ATN(X) + 1.570796
HYPERBOLIC	
SINE	(EXP(X) − EXP(−X))/2
COSINE	(EXP(X) + EXP(−X))/2
TANGENT	−EXP(−X)/(EXP(X) + EXP(−X))*2 + 1
SECANT	2/(EXP(X) + EXP(−X))
COSECANT	2/(EXP(X) − EXP(−X))
COTANGENT	EXP(−X)/(EXP(X) − EXP(−X))*2 + 1
INVERSE HYPERBOLIC	
SINE	LOG(X + SQR(X*X + 1))
COSINE	LOG(X + SQR(X*X − 1))
TANGENT	LOG((1 +X)/(1 −X))/2
SECANT	LOG((SQR(−X*X + 1) + 1)/X)
COSECANT	LOG((SGN(X)*SQR(X*X + 1) + 1)/X)
COTANGENT	LOG((X + 1)/(X − 1))/2

DEC	HEX	GET KEY	UPPER CASE PRINT CHR$	UPPER CASE POKE	LOWER CASE PRINT CHR$	LOWER CASE POKE	TOKEN OR ASCII
0	00			@		@	NUL
1	01			A		a	SOH
2	02			B		b	STX
3	03			C		c	ETX
4	04			D		d	EOT
5	05	©2	white	E	white	e	ENQ
6	06	©←		F		f	ACK
7	07			G		g	BEL
8	08		toggle off	H	toggle off	h	BS
9	09		toggle on	I	toggle on	i	HT
10	0A			J		j	LF
11	0B			K		k	VT
12	0C			L		l	FF
13	0D	RETURN	curs next lin	M	curs next lin	m	CR
14	0E		set low case	N	set low case	n	SO
15	0F			O		o	SI
16	10			P		p	DLE
17	11	CUR↓	cursor down	Q	cursor down	q	DC1
18	12	©9 ©R	reverse on	R	reverse on	r	DC2
19	13	HOME	home cursor	S	home cursor	s	DC3
20	14	DEL	delete char	T	delete char	t	DC4
21	15			U		u	NAK
22	16			V		v	SYN
23	17			W		w	ETB
24	18			X		x	CAN
25	19			Y		y	EM
26	1A			Z		z	SUB
27	1B			[		[	ESC
28	1C	©3	red	£	red	£	FS
29	1D	CUR→	cursor right	]	cursor right	]	GS
30	1E	©6	green	↑	green	↑	RS
31	1F	©7	blue	←	blue	←	US
32	20	space					space
33	21	!	!	!	!	!	!
34	22	”	"	"	"	"	”
35	23	#	#	#	#	#	#
36	24	$	$	$	$	$	$
37	25	%	%	%	%	%	%
38	26	&	&	&	&	&	&
39	27	’	’	’	’	’	’
40	28	(	(	(	(	(	(
41	29	)	)	)	)	)	)
42	2A	*	*	*	*	*	*
43	2B	+	+	+	+	+	+
44	2C	,	,	,	,	,	,
45	2D	−	−	−	−	−	−
46	2E	.	.	.	.	.	.
47	2F	/	/	/	/	/	/
48	30	0 ⊗0	0	0	0	0	0
49	31	1	1	1	1	1	1
50	32	2	2	2	2	2	2
51	33	3	3	3	3	3	3
52	34	4	4	4	4	4	4
53	35	5	5	5	5	5	5
54	36	6	6	6	6	6	6
55	37	7	7	7	7	7	7
56	38	8	8	8	8	8	8
57	39	9	9	9	9	9	9
58	3A	:	:	:	:	:	:
59	3B	;	;	;	;	;	;
60	3C	<	<	<	<	<	<
61	3D	=	=	=	=	=	=
62	3E	>	>	>	>	>	>
63	3F	?	?	?	?	?	?

BASIC COMMANDS	
EXAMPLE	**DESCRIPTION**
CONT	**CONT**inue execution after break, **STOP**, or **END.**
LIST	Display all **BASIC** program lines.
LIST 10	Display **BASIC** program line 10.
LIST 10—100	Display **BASIC** lines 10 thru 100.
LIST 200—	Display **BASIC** lines 200 thru end of program.
LIST —150	Display **BASIC** lines 0 thru 150.
LOAD	Load first program on tape into memory. (When used inside **BASIC** program, executes at once.
LOAD "TEST"	Load program named **TEST** into memory from tape.
LOAD "TEST",dev	Load program named **TEST** into memory from device. (dev = 1 for tape, 8 for disk).
LOAD "TEST",dev,1	Load program into same location saved from.
LOAD " * ",8	Load first program on disk into memory.
LOAD "TE * ",8	Load first program, prefix TE to memory from disk.
LOAD A$	Load program named in variable A$ from tape.
NEW	Clear current program out of memory.
RUN	Execute **BASIC** program.
RUN 1000	Execute **BASIC** program starting at line 1000.
SAVE	Save program in memory onto tape without name.
SAVE "TEST"	Save program onto tape, and name it **TEST.**
SAVE A$	Save program onto tape, using name in variable A$.
SAVE A$,dev	Save program onto device (1-tape, 8-disk).
SAVE "TEST",dev,opt	Save program onto device with option: 1—Record program's address in memory for loading. 2—Place an end-of-tape record after the program. 3—Do both options 1 and 2 above.
VERIFY	Compare first program on tape against memory program.
VERIFY "TEST"	Compare tape program **TEST** against one in memory.
VERIFY "TEST",8	Compare disk program **TEST** against one in memory.

BASIC STATEMENTS [ ] = Optional
STATEMENT • DESCRIPTION
CLOSE fileno • Close open file **fileno.**
CLR • Clear vars & arrays, and **RESTORE** pointer to **DATA** table.
CMD fileno • Route screen output to **fileno** until **PRINT#** and **CLOSE.**
DATA [item [,item . . .] • Store **item**s into data memory table.
DEF FNx(arg)=expression • Define user function named **FNx.**
DIM var (n1 [,n2, . . .]) [,var1(n3,n4, . . .)] • Define array storage.
END • Halt program execution until **CONT.**
FOR var=num1 TO num2 [STEP incr] • Loop thru **NEXT** until **var** greater **num2.**
GET var$** • Scan keyboard and place key pressed into **var$.
GET# fileno, var • Read byte from **fileno** into **var.**
GOSUB lineno • Perform routine at **lineno** until **RETURN.**
GOTO lineno • Branch or transfer control to **lineno.**
IF cond THEN lineno • If **cond**ition met then branch to **lineno.**
IF cond THEN action [:action . . .] • If **cond**ition met then do **action**(s).
INPUT [msg;] var [,var . . .] • Prompt for and read keyed response.
INPUT# fileno, var [,var . . .] • Read data from **fileno** into **var**(s).
[LET] var=value • Assign a **value** to a **var**iable.
NEXT [var [,var . . .] • Identify end of **FOR** loop(s).
ON num GOSUB line [,line . . .] • Perform routine **num** of **line** list.
ON num GOTO line [,line . . .] • Branch to lineno **num** of **line** list.
OPEN fileno **[,dev] [,option] [,data]** • **Open fileno on device dev using option and filename/param data.**
POKE addr,value • Move **value** (0—255) to **addr** in memory.
PRINT item [; [, [item . . .] • Display **item**(s) on screen. (TAB & SPC useable)
PRINT# fileno,item [; [, [item . . .] • Write **item**(s) onto **fileno.**
READ var [,var . . .] • Read **DATA** from memory table.
REM • **REM**arks follow this statement.
RESTORE • Reset pointer to start of **DATA** memory table.
RETURN • Return from **GOSUB.**
STOP • Halt program execution until **CONT.**
SYS addr • Branch to machine language routine at **addr.**
TI$=num • Set clock timer (HHMMSS). Hours, minutes, seconds.
WAIT addr, and1 [,xor2] • Pause execution until data at **addr** matches arg.

Published 1983 by Nanos Systems Corp., P.O. Box 24344, Speedway, IN 46224
(317) 244 - 4078

Printed in U.S.A.